d

TOne Security Document

Contents

[Document Purpose 3](#_Toc422830756)

[Security Module Scope 4](#_Toc422830757)

[Out of Scope 5](#_Toc422830758)

[Security Design 6](#_Toc422830759)

[Business Entities Structure 6](#_Toc422830760)

[Permissions 7](#_Toc422830761)

[Groups 7](#_Toc422830762)

[Group Members 7](#_Toc422830763)

[Group Permissions 7](#_Toc422830764)

[Authentication 9](#_Toc422830765)

[Authorization 11](#_Toc422830766)

[Configuring Authorization 11](#_Toc422830767)

[System Entities 12](#_Toc422830768)

[Assets (Views and UI Controls) 13](#_Toc422830769)

[Views 13](#_Toc422830770)

[UI Controls 14](#_Toc422830771)

[Assign Permissions to Holders 14](#_Toc422830772)

[Security Module in One Picture 15](#_Toc422830773)

[Assigning Permissions in Details 16](#_Toc422830774)

[Permissions Inheritance 16](#_Toc422830775)

[Full Control 17](#_Toc422830776)

[Group Inheritance 17](#_Toc422830777)

[Deny Access 17](#_Toc422830778)

[Break Inheritance 18](#_Toc422830779)

[Menu Items Trimming 19](#_Toc422830780)

[Audience 20](#_Toc422830781)

[UI Controls Trimming 21](#_Toc422830782)

# Document Purpose

The purpose of this document is to discuss the security implementation of TOne. This is composed of:

* Authentication
* Authorization

# Security Module Scope

This security module currently covers the following subjects:

* **Users and Groups**: management screens for Users and Group of users
* **Authentication**: in its basics, a login screen with a token created after validating credentials. The token is currently saved into a browser cookie.
* **Authorization**:
  + **System Entities**: configuring system entities and linking them to Views and UI Controls.
    - **Groups Inheritance**
    - **Module/Entities Inheritance**
    - **Full Control, Deny and Break inheritance logic**
  + **Menu Items** **trimming**: trimming the menu items according to the assigned permissions give to a logged in user.
  + **UI Controls trimming**: trimming the UI Controls according to the assigned permissions given to a logged in user.
  + **Audience**: another level of UI trimming on the menu items level

# Out of Scope

Below is a list of the missing subjects that needs to be supported in the security module:

* **Users**:
  + Setting encrypted random passwords on user creation (now it is set automatically to 1)
  + Support for resetting passwords
  + Check the list of Groups the user is member of
* **Authentication**:
  + Handling the cases of Login Failure
  + Validating the given password on login using an encryption logic (now it is set as plain text)
  + Adding more information to the created token
  + Make the token expires after a configured period of time
  + Validate token server side using an encryption logic to consider a user authenticated (now the validation if the user is authenticated is done only client side, checking if there is a cookie created)
* **Authorization**:
  + **Web API Security**: apply security on controllers “server side” (what is done now is only UI trimming on the client side according to the permissions given for a logged in user).
  + **Business Categories**: group users on creation according to certain categories (Financial, Technical), and take these groups into consideration when resolving permissions
  + Data Security: apply security on the data retrieved from the server considering the data as an asset bound to system modules/entities.

# Security Design

The scope of this section is to discuss the design of the authorization part of the security. The authorization consists of allowing an authenticated user to ONLY do what he is allowed to do while using TOne UI (or API). It is summarized in the following statements:

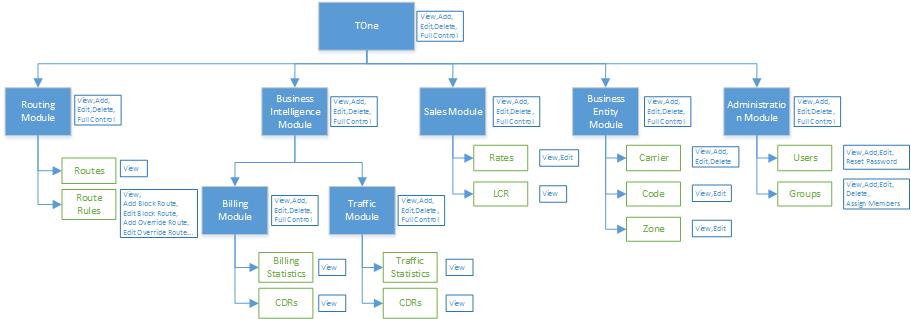
* An authenticated user can access ONLY TOne pages that he/she has permissions to view
* An authenticated user can see ONLY data that he/she has permissions to view
* An authenticated user can perform ONLY actions that he/she has permissions to perform

In order to implement the above requirements in a flexible manner, we would:

* Define a structure for the system business entities in a way that allows the system administrator(s) to configure permissions easily
* To implement user groups membership which also gives more flexibility in configuring permissions

## Business Entities Structure

TOne Business Entities would be structured in a hierarchical manner:



The above diagram shows an example business entities hierarchy. In this case, TOne would be composed of multiple modules (e.g. Routing, Business Intelligence…). Each module would also have multiple modules and/or entities (e.g. Routes, Route Rules, Rates…).

Each entity in the hierarchy would defines the types of actions it has. For example:

* The Route Rules entity defines the following actions:
  + View
  + Add Block Route
  + Edit Block Route
  + Add Override Route
  + Edit Override Route
  + …
* The Users entity defines the following actions:
  + View
  + Add
  + Edit
  + Reset Password

And each module in the hierarchy defines the following actions:

* View
* Add
* Edit
* Delete
* Full Control

## Permissions

A permission is composed of four parts:

* Permission Holder: User or Group
* Secure Resource: Business Entity or Module (as discussed in previous section)
* Action: e.g. View, Add, Edit…
* Permission Flag: Allow or Deny

The following rules would be also applied:

* The business entities inherit the permissions assigned on their parent modules in the Business Entities hierarchy
* It would be possible to stop Permission Inheritance on any Module/Entity in the hierarchy
* The Deny Permission Flag overrides any Allow Permission Flag

## Groups

As discussed earlier, the system will give ability to create user Groups to make security configuration easier. The Group is composed of two parts:

* Members: this part would give a way or more to tell which users are members of this Group
* Permission: this part would give a way or more to tell what permissions this Group has and to which business entities

### Group Members

Group Members would be assigned in two ways:

* Explicitly: the administrator sets the users that are the members of the group
* Implicitly: the members of the group would be defined implicitly this way:
  + Each user would have a category attribute (e.g. Financial, Technical, Administration…)
  + The administrator would then configure the group to include as members the users that have specific category(ies)

### Group Permissions

Group Permissions would be also assigned in two ways:

* Explicitly: the administrator sets the permissions of the group directly on the Business Entity hierarchy
* Implicitly: the permissions of the group would be defined implicitly this way:
  + Each Module/Entity would have a category and sub-category attributes (e.g. Financial, Technical, Billing, Traffic…)
  + The administrator would then configure the permissions of the group by assigning permissions on the modules/entities that have specific category(ies) and sub-category(ies)

# Authentication

Each user wants to access the system needs to be authenticated. A user name and password are assigned to each user and saved into the Users Table.

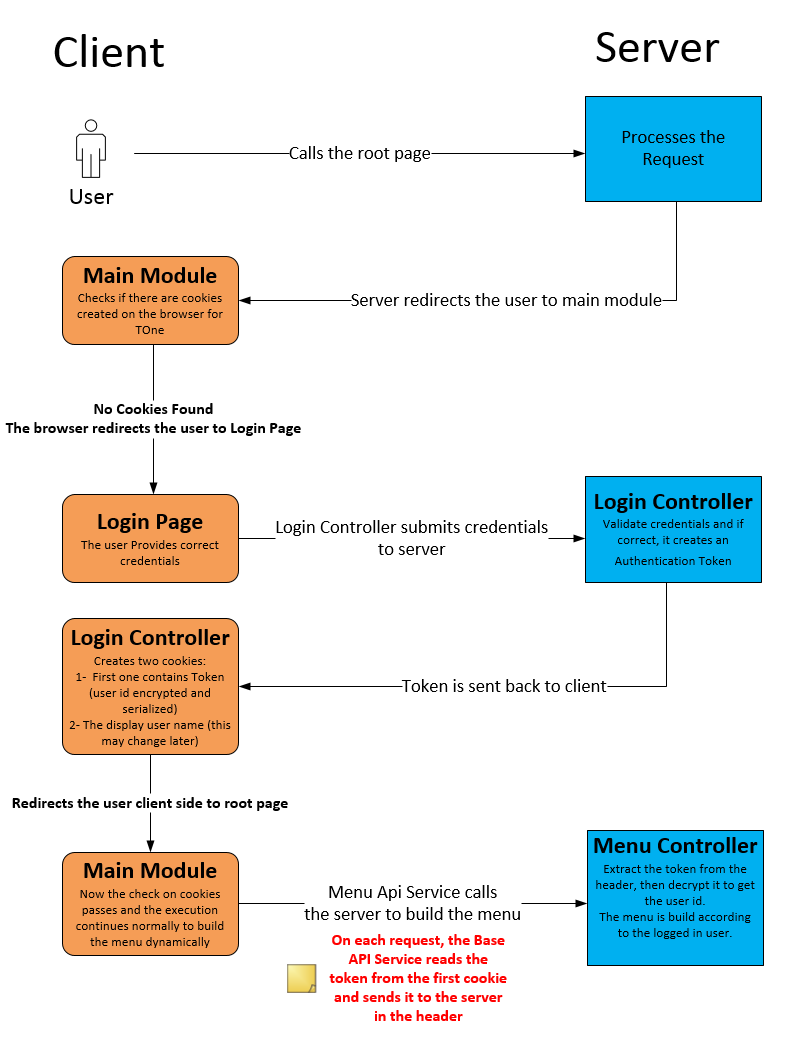
When navigated to the root page, if the user is not authenticated the application redirects him back to login page.

After providing the correct credentials, the server creates a Token and sends it back to the client. The client saves this token into a cookie for later requests.

When the user requests the root page again, if these cookies are found the user is considered authenticated. Then, the token is added to the header of the request and submitted to the server.

***Note: authentication is not completed yet; server side authentication should happen also to make sure the token is valid. The token should also expires and renewed on the browser using the saved credentials.***

Below is a sequence diagram that shows the client-server communications during the current authentication process.



# Authorization

As discussed earlier in this document, business entities (or system entities) are pre-configured for the system and rendered as a business entities hierarchical tree.

This tree classifies the business aspects of the system into modules and entities. Each business module or entity has its own permission flags (View, Add, Edit…).

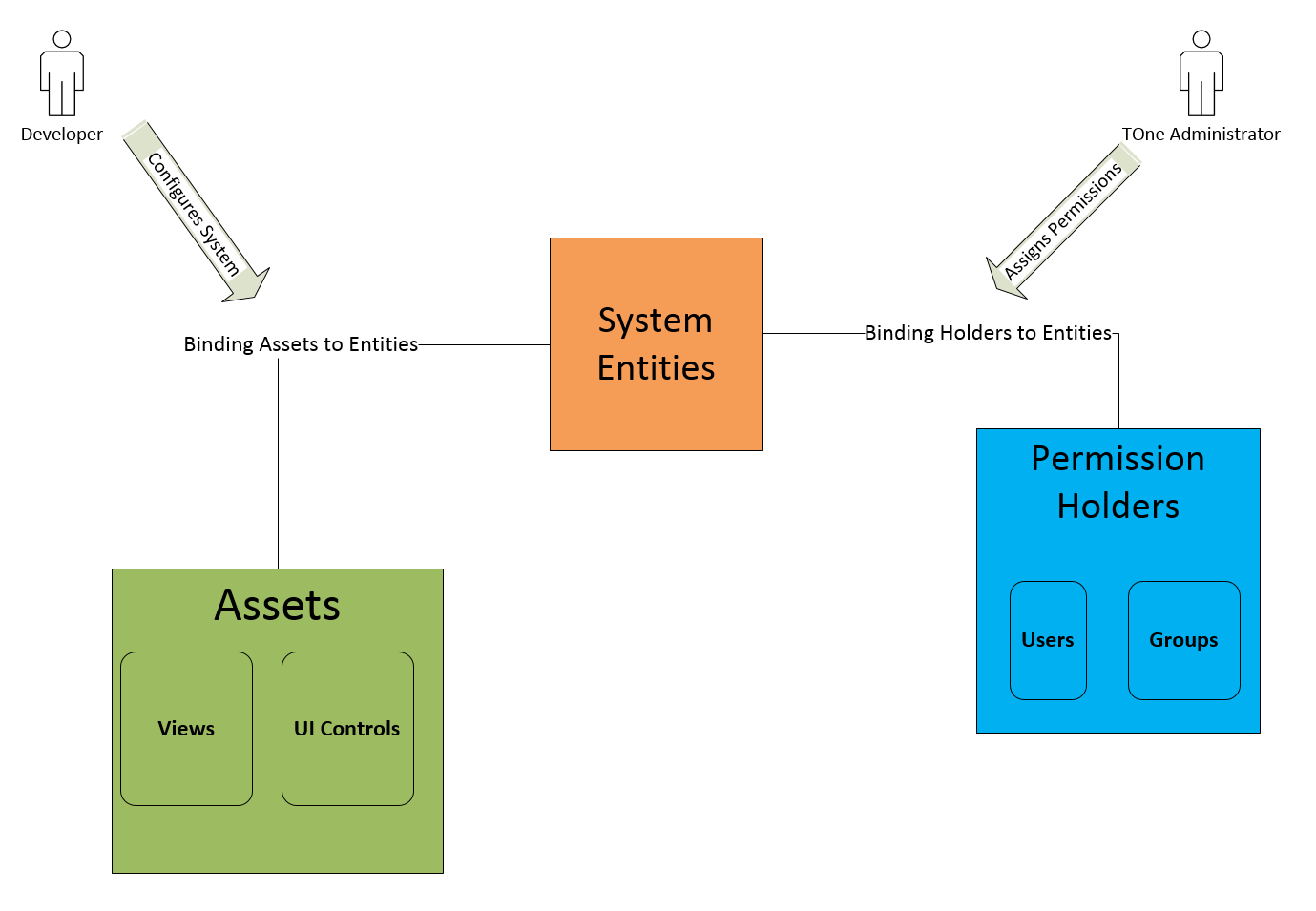
The authorization actors can be classified into three groups:

1. **System Entities**: the pre-configured business entity hierarchy.
2. **Assets**: the parts of the system that are subject for authorization (Pages, UI Controls…)
3. **Permission Holders**: the clients requesting access to system Assets (Users, Groups…)

## Configuring Authorization

Configuring authorization in this module is divided into two main steps:

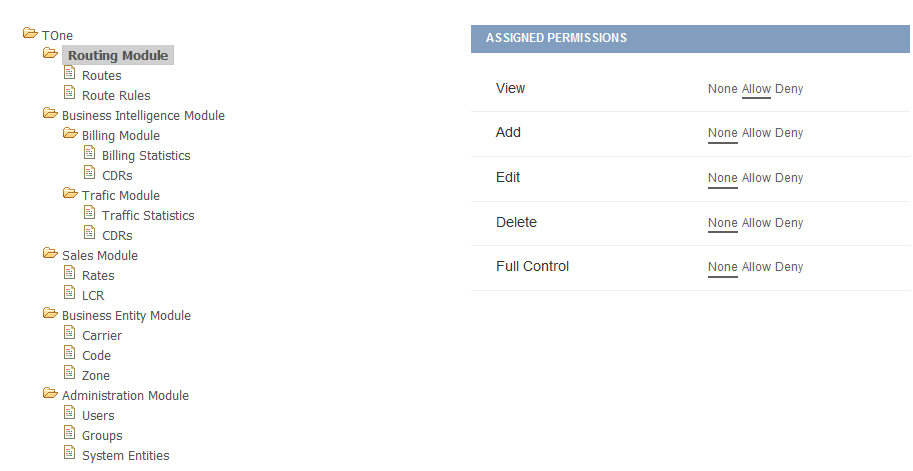
1. Through the development process, developers bind the different assets into their respective business modules or entities.
2. At runtime, administrators can assign permission holders access to specific business entities flags.



## System Entities

The pre-configured business modules and entities are loaded into one hierarchy and rendered on the user interface as a Tree.

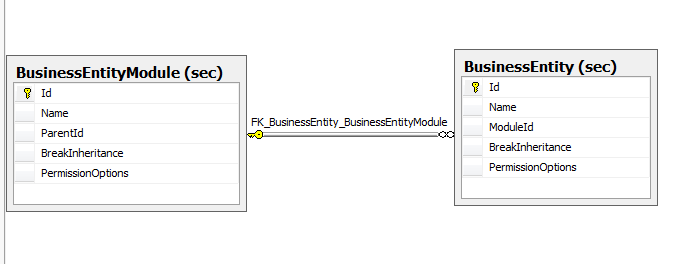
Below is a screen shot showing that tree:



Each node in the tree can be a Module or an Entity. Each node can have different Permission Flags (View, Add, Edit…). Permission flags are serialized into a JSON string and saved into the database.

Two tables are responsible about building the above tree:

1. sec.BusinesEntityModule
2. sec.BusinessEntity



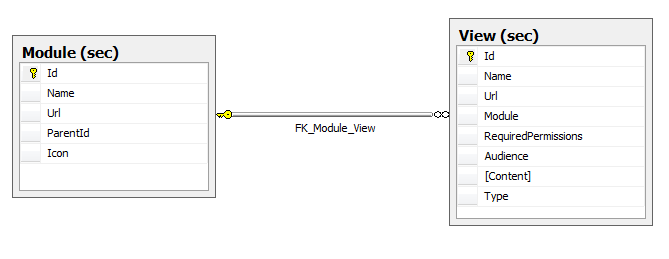
## Assets (Views and UI Controls)

### Views

Views (or pages in other words) are saved into the database in one table called sec.View. This table has information about each page (Name, URL…) and are groups logically into modules.

The menu on the user interface is built dynamically by reading from the below two tables:

1. sec.Module
2. sec.View



Permissions are set to Views and not modules. If all views are trimmed for a user, their parent module is not shown in the user interface menu.

Setting the “Required Permissions” column to null means that this view (page) is public for all users.

The required permissions are set by developers during development time in the following manner:



This means, that the page called “Users” is only displayed for holders with **allow** permissions set to “**Users Entity**” => **View** Flag.

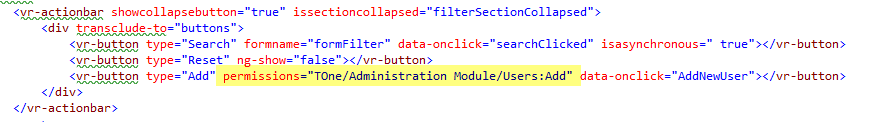
The Users Entity is found in the tree under: **TOne/Administration Module**

A page may require two permissions to be shown and more than one permission flag for each:

**TOne/Administration Module/Users: View, Add, Edit | TOne/Administration Module/Groups: View**

### UI Controls

Same applies for UI controls. Each control that has to be restricted and not show for public needs to have the permissions attribute set as follow:

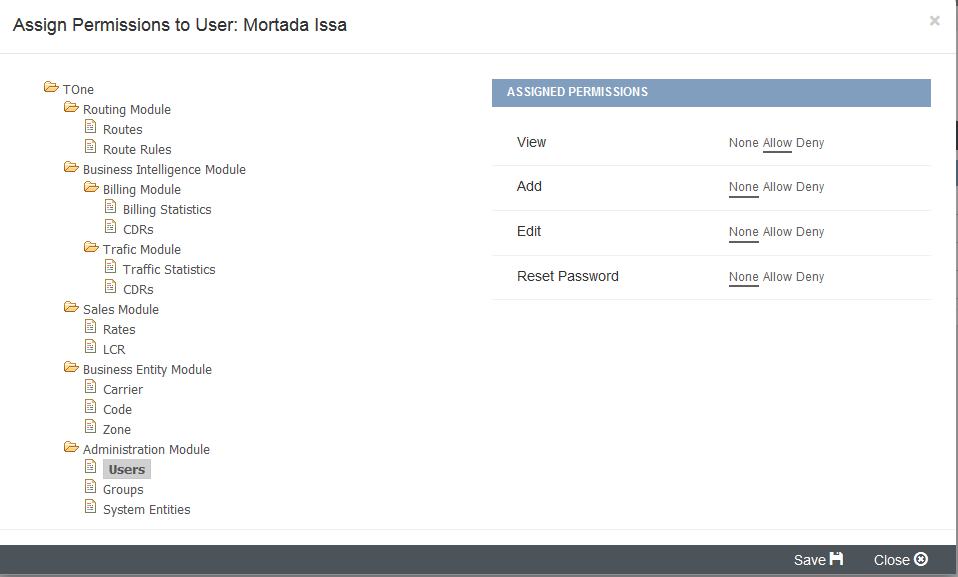


This means that the button Add in Users Page is only shown for holders with **allow** permissions set to “**Users Entity**” => **Add** Flag.

## Assign Permissions to Holders

After setting up the system entity tree and binding Views – UI Controls to their respective entities, the application is published and users can start checking UI Trimming in Action.

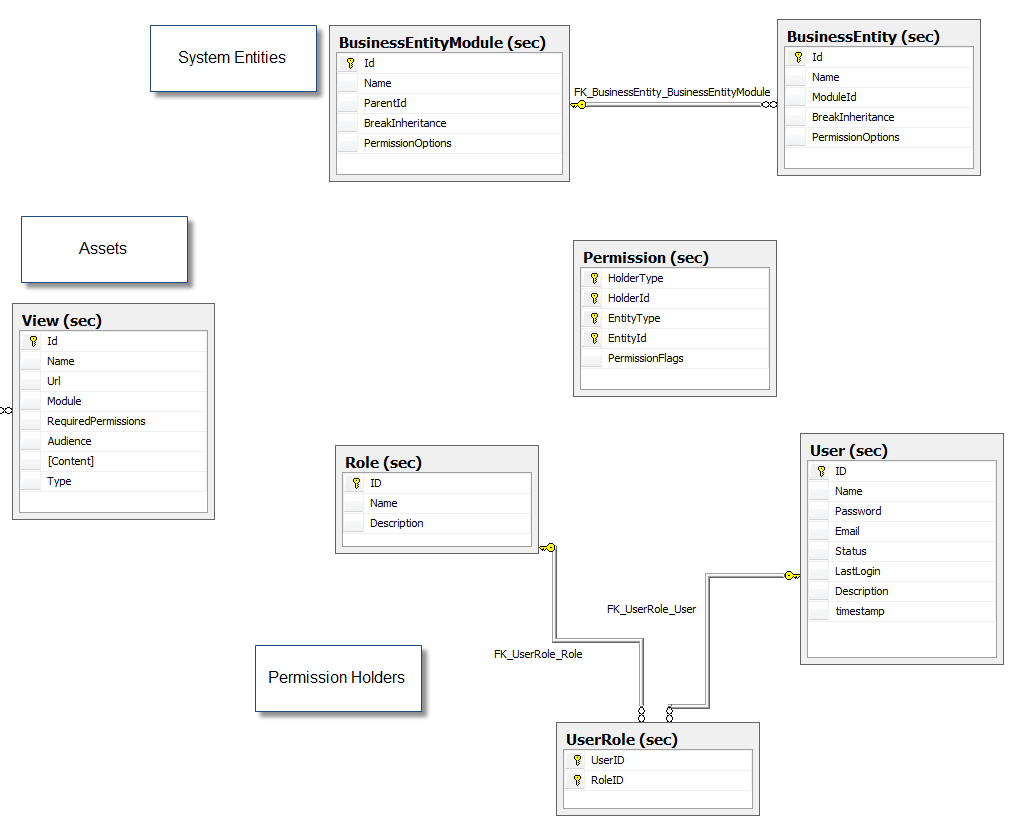
At runtime, the administrator can go to Users page and start assigning permissions to different users as shown in the following screen shot:



The above means that Mortada Issa is only allowed to see the Users page, but cannot see the Add button on this page.

Unless the administrator sets the permission flag “Add” to Allow, Mortada Issa will not see the Add button on Users Page. At the same time, if “View” permission flag is set to none. Mortada Issa will not be able to see the Users page at all.

## Security Module in One Picture



The above picture shows the Diagram of all responsible tables after integrating the security module into TOne.

In each product you want to integrate the security module into, you should have Assets, Permission Holders and you should define your system entities.

Permissions that are set by administrators are serialized and saved into the PermissionFlags column of Permission table.

Each record in the permission table goes to a combination of:

1. Holder: can be a user or a role in this case
2. Entity: can be a module or an entity
3. Flags: can be saved in the following format (View – Allow, Add – Allow, Edit – Deny).

*Note: Permission Flags set to “None” are not saved into the permissions flag column.*

## Assigning Permissions in Details

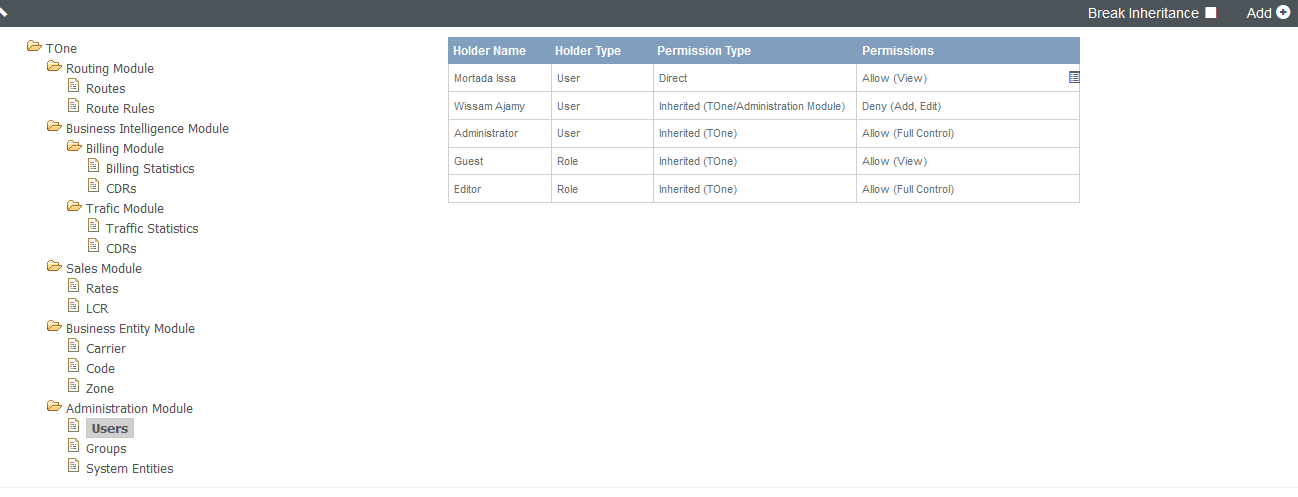
In TOne, you can assign permission to holders from three places:

1. Users Grid
2. Groups Grid
3. System Entities Page

Assigning a permission to a single user or a single group can be effective for some cases, but in most cases, the administrator needs to look at permissions from a different perspective.

System Entities page serves this purpose and shows the administrator the permissions given to a specific system module/entity.

Below is a screen shot showing the permissions assigned to Users entity:



Discussing the permissions given on Entity “Users”:

**Mortada Issa**: this is a normal user that is given View access to “Users” page, discussing each column:

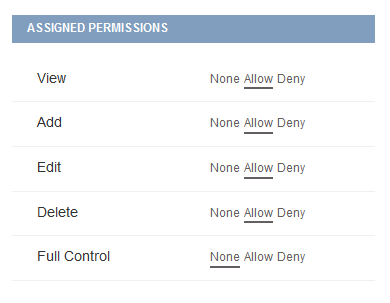
1. **Holder Type**: this is set to “User” and can be a Group/Role in other cases.
2. **Permission Type**: this column differentiates between permissions assigned directly to this Entity (called Direct) and the ones inherited from other modules (called Inherited). The Permission Type column also shows the path to the module from where this permission is inherited.
3. **Permissions**: it shows the list of Permission Flags given for this Holder, each with the Flag value (Allow/Deny).

### Permissions Inheritance

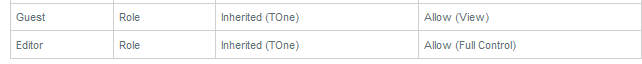
All permissions given to parent module(s) affect the permissions given for an Entity. Taking an example the Administrator in the previous screen shot who is given Full Control to TOne System Module, this means that the Administrator can see any page or a UI control that requires any permission flag (View, Add, Edit…) on any system Module/Entity (Code, Zone, Users, Groups…) under TOne.

### Full Control

On a certain module and instead of setting the values of all Flags to “Allow/Deny”, you can set the value of Full Control permission Flag. Full Control only applies for System Modules and not Entities.



### Group Inheritance



In the previous example, you can see Holders of Type Group/Role. The screen shot shows inherited permissions on the “Users” entity coming from the root Module “TOne”.

This means that all members who belong to “Guest” group have View access to this Entity “Users”. Same applies for members of the “Editor” group who have Full Control on all modules under “TOne”.

### Deny Access

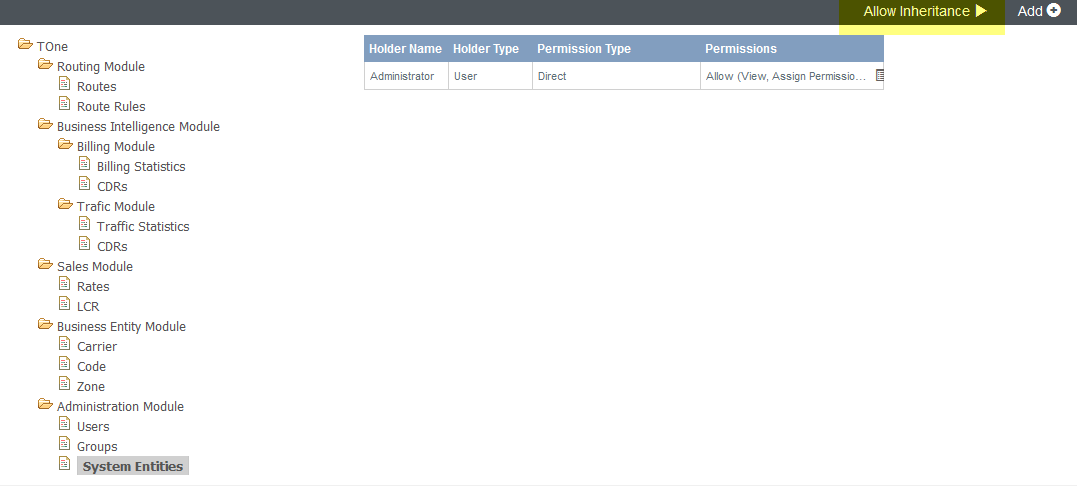


Take an example Wissam Ajamy who is member of “Editor” group. According to group inheritance Wissam has Full Control to “Users” Entity.

Wissam also has “Add, Edit” Permission Flags set to Deny on “TOne/Administration Module”. This restricts adding or editing on all entities under this module (Users, Groups and System Entities).

### Break Inheritance



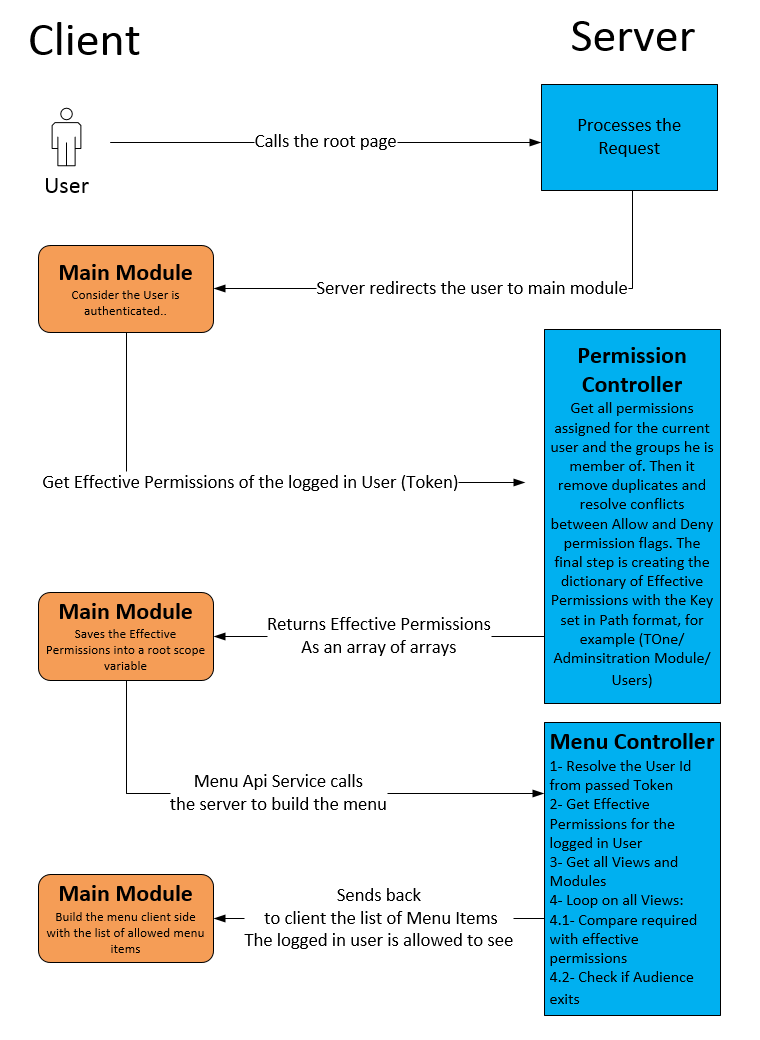


The above screen shots show the permissions assigned to two entities:

1. **Users**: where inheritance is allowed. The administrator can choose to Break Inheritance.
2. **System Entities**: on this module, inheritance is broken. Even though “Editor” group is given “Full Control” to all modules under “TOne”, only the User Administrator is allowed to [View, Assign Permissions] on “System Entities” entity.

Notice that only Direct Permissions appear for System Entities in the permissions grid. The administrator can allow inheritance again on this module.

## Menu Items Trimming



### Audience

In the previous screen shot, in the last steps the server starts checking if the user is allowed to see each view.

The first check is done on the level of security by comparing the required permissions given on each view with the list of effective permissions given for the logged in user.

Another level of UI Trimming is applied which is called Audience. Each view can have a list of users and/or groups as audience. Unless the logged in user is in the audience list, he cannot see the view in the menu.

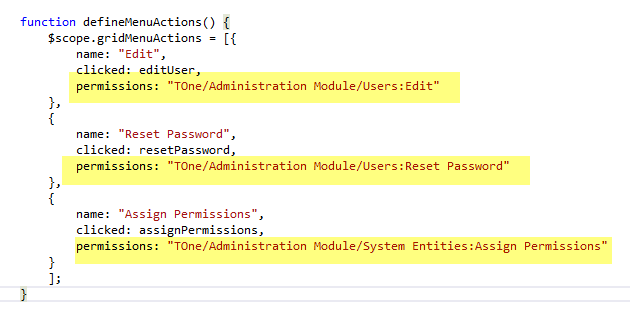
### UI Controls Trimming

Each directive has its permissions attribute that needs to be set as follow:

**Button Directive:**



**Grid Menu Actions:**



*Note: until the time this document is written, not all directives supports permissions attribute.*

Below is the process of checking the permissions on each directive:

